Q1: What are the advantages of Polymorphism?

- Code Reusability: By defining a superclass with common behaviors and letting subclasses override them, you can reuse superclass methods in subclasses without duplication.

- Flexibility and Extensibility: Code becomes capable of working with objects of multiple types, simplifying extension and modification without altering existing code.

- Encapsulation Support: It aids in achieving encapsulation, concealing object implementation details and exposing only interfaces for easier code management.

- Dynamic Binding: Enables runtime resolution of method calls based on object type, rather than compile-time, ideal for scenarios involving diverse objects through a common interface or superclass.

- Code Clarity and Maintainability: Allows writing code operating on objects at a higher abstraction level, enhancing clarity, understanding, and ease of maintenance.

Q2: How is Inheritance useful to achieve Polymorphism in Java?

- Polymorphism in Java entails executing a single action in different ways. Inheritance is important because of allowing one class to acquire the properties and attributes of the parent class. It also enables the use of inheriting properties to perform varied tasks, thus achieving the same action in multiple methods.

Q3: What are the differences between Polymorphism and Inheritance in Java?

- In Java, inheritance allows a class (subclass) to inherit properties and behaviors from another class (superclass), promoting code reuse. Polymorphism, on the other hand, enables objects of different classes to be treated as objects of a common superclass, allowing methods to be invoked dynamically based on the actual object type. Inheritance is a mechanism for class hierarchy, while polymorphism allows for flexible method invocation.